Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade

Jiatao Gu*
Facebook AI Research
jgu@fb.com

Xiang Kong*
Language Technologies Institute
Carnegie Mellon University
xiangk@cs.cmu.edu

Abstract

Fully non-autoregressive neural machine translation (NAT) simultaneously predicts tokens with single forward of neural networks, which significantly reduces the inference latency at the expense of quality drop compared to the Transformer baseline. In this work, we target on closing the performance gap while maintaining the latency advantage. We first inspect the fundamental issues of fully NAT models, and adopt *dependency reduction* in the learning space of output tokens as the primary guidance. Then, we revisit methods in four different aspects that have been proven effective for improving NAT models, and carefully combine these techniques with necessary modifications. Our extensive experiments on three translation benchmarks show that the proposed system achieves the state-of-the-art results for fully NAT models, and obtains comparable performance with the autoregressive and iterative NAT systems. For instance, one of the proposed models achieves **27.49** BLEU points on WMT14 En-De with **16.5**× speed-up compared to similar sized autoregressive baseline under the same inference condition. The implementation of our model is available here[1].

1 Introduction

State-of-the-art neural machine translation (NMT) systems are based on autoregressive models (Bahdanau et al., 2015; Vaswani et al., 2017) where each generation step depends on the previously generated tokens. This sequential nature inevitably leads to inherent latency at inference time. On the other hand, non-autoregressive neural machine translation models (NAT, Gu et al., 2018a) attempt to generate output sequences in parallel to speed-up

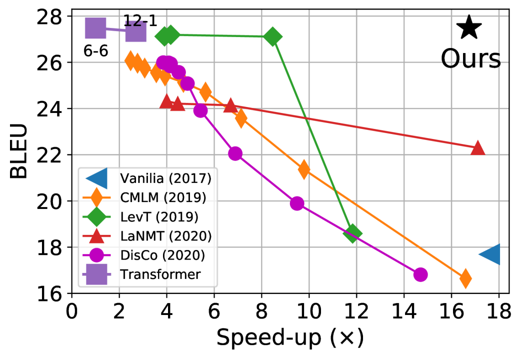

Figure 1: The translation quality v.s. inference speed-up on WMT'14 En→De test set. The upper right corner achieves the best trade-off.

the decoding process. The incorrect independence assumption nevertheless prevents NAT models to properly learn the dependency between target tokens in real data distribution, resulting in poorer performance compared to autoregressive (AT) models. One popular solution to improve the NAT translation accuracy is to sacrifice the speed-up by incorporating an iterative refinement process, through which the model explicitly learns the conditional distribution over partially observed reference tokens (Ghazvininejad et al., 2019; Gu et al., 2019). However, recent studies (Kasai et al., 2020b) indicated that iterative NAT models seem to lose the speed advantage compared to AT models with careful tuning of the layer allocation. For instance, an AT model with *deep encoder and shallow decoder* obtains similar latency as iterative NAT models without hurting the translation accuracy.

Therefore, how to build a competitive fully NAT model without iterative refinements calls for more exploration. Several works (Ghazvininejad et al., 2020a; Saharia et al., 2020; Qian et al., 2020) have recently been proposed to improve the training of NAT, though the performance gap compared to the iterative ones remains. In this work, we first ar-

* Equal contribution.
[1] https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 120–133
August 1–6, 2021. ©2021 Association for Computational Linguistics

gue that the key to successfully training a fully NAT model is to perform *dependency reduction* in the learning space of output tokens (§ 2) from all aspects. With this guidance, we revisit various methods which are able to reduce the dependencies among target tokens as much as possible including four different perspectives, i.e., training corpus (§ 3.1), model architecture (§ 3.2), training objective (§ 3.3) and learning strategy (§ 3.4). The performance gap can not be near closed unless we combine these techniques' advantages.

We validate the proposed fully NAT model on standard translation benchmarks including 5 translation directions where our system achieves new state-of-the-art results for fully NAT models on all directions. We also demonstrate the quality-speed trade-off comparing with AT and recent iterative NAT models in Figure 1. Moreover, compared to the Transformer baseline, our model achieves **16.5**$\times$ inference speed-up under the same software/hardware conditions while maintaining comparable translation quality.

2 Motivation

Given an input sequence $\boldsymbol{x} = x_1 \ldots x_{T'}$, an autoregressive model (Bahdanau et al., 2015; Vaswani et al., 2017) predicts the target $\boldsymbol{y} = y_1 \ldots y_T$ sequentially based on the conditional distribution $p(y_t|y_{<t}, x_{1:T'}; \theta)$, which tends to suffer from high latency in generation especially for long sequences. In contrast, non-autoregressive machine translation (NAT, Gu et al., 2018a), proposed for speeding-up the inference by generating all the tokens in parallel, has recently been on trend due to its nature of parallelizable on devices such as GPUs and TPUs. A typical NAT system assumes a conditional independence in the output token space, that is

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{t=1}^{T} \log p_\theta(y_t|x_{1:T'}) \qquad (1)$$

where θ is the parameters of the model. Typically, NAT models are modeled with Transformer without causal attention map in the decoder side. As noted in Gu et al. (2018a), the independence assumption, however, generally does not hold in real data distribution for sequence generation tasks such as machine translation (Ren et al., 2020), where the failure of capturing such dependency between target tokens leads to a serious performance degradation in NAT. This is a fairly understandable but fundamental issue of NAT modeling which can be easily shown with a toy example in Figure 2.

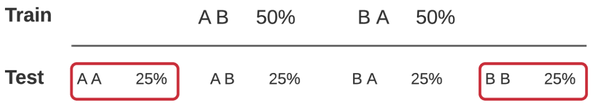

Figure 2: Toy example shows that NAT fails to learn when dependency exists in output space.

Given a simple corpus with only two examples: *AB* and *BA*, each of which has 50% chances to appear. It is designed to represent the dependency that symbol *A* and *B* should co-occur. Although such simple distribution can be instantly captured by any autoregressive model, learning the vanilla NAT model with maximum likelihood estimation (MLE, Eq. (1)) assigns probability mess to incorrect outputs (*AA*, *BB*) even these samples never appear during training. In practice, the dependency in real translation corpus is much more complicated. As shown in Figure 1, despite the inference speed-up, the vanilla NAT leads to a quality drop over **10** BLEU points.

To ease the modeling difficulty, recent state-of-the-art NAT systems (Lee et al., 2018; Stern et al., 2019; Ghazvininejad et al., 2019; Gu et al., 2019; Kasai et al., 2020a; Shu et al., 2020; Saharia et al., 2020) trade accuracy with latency by incorporating iterative refinement in non-autoregressive prediction. For instance, Gu et al. (2019) learns to translate by editing (deletion, insertion) on previously generated sequence iteratively. Although iterative NAT models have already achieved comparable or even better performance than the autoregressive counterpart, Kasai et al. (2020b) showed AT models with a deep encoder and a shallow decoder can readily outperform strong iterative models with similar latency, indicating that the latency advantage of iterative NAT has been overestimated.

By contrast, while maintaining a clear speed advantage, fully NAT system – model makes parallel predictions with single neural network forward – still lags behind in translation quality and has not been fully explored in literature (Libovický and Helcl, 2018; Li et al., 2018; Sun et al., 2019; Ma et al., 2019; Ghazvininejad et al., 2020a). This motivates us in this work to investigate various approaches to push the limits of learning a fully NAT model towards autoregressive models regardless of the architecture choices (Kasai et al., 2020b).

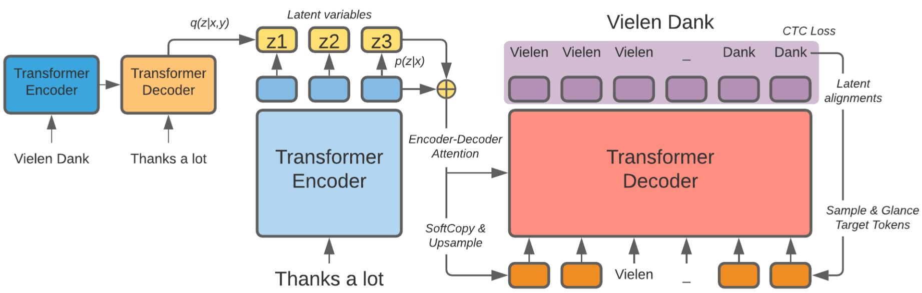

Figure 3: The overall framework of our fully NAT model.

3 Methods

In this section, we discuss several essential ingredients to train a fully NAT model. As discussed in § 2, we argue that the guiding principle of designing any NAT models is to perform *dependency reduction* as much as possible in the output space so that it can be captured by the NAT model. For example, iterative-based models (Ghazvininejad et al., 2019) explicitly reduce the dependencies between output tokens by learning the conditional distribution over the observed reference tokens. The overall framework of training our fully NAT system is presented in Figure 3. We also summarize the pros/cons for each proposed method in Table 1 for reference.

3.1 Data: Knowledge Distillation

The most effective *dependency reduction* technique is knowledge distillation (KD) (Hinton et al., 2015; Kim and Rush, 2016) which is firstly proposed to improve NAT in Gu et al. (2018a) and has been widely employed for all subsequent NAT models. The original target samples are replaced with sentences generated from a pre-trained autoregressive model. As analyzed in Zhou et al. (2020), KD is able to simplify the training data where the generated targets have less noise and are aligned to the inputs more deterministically. Also, it showed that the capacity of the teacher model should be constrained to match the desired NAT model to avoid further degradation, especially for weak NAT students without iterative refinement.

3.2 Model: Latent Variables

Different from iterative NAT, *dependency reduction* can be done with (nearly) zero additional cost at inference by adding latent variables to the model. In such case, output tokens $y_{1:T}$ are modeled conditionally independent over the latent variables $\boldsymbol{z}$ which are predicted from the prior distribution:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \log \int_{\boldsymbol{z}} p_\theta(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x}) d\boldsymbol{z} \quad (2)$$

$\boldsymbol{z}$ can be either extracted by a fixed external library (e.g. fertility in Gu et al. (2018a)), or jointly optimized with the NAT model using variational autoencoders (VAEs) (Kaiser et al., 2018; Shu et al., 2020) or normalizing flow (Ma et al., 2019).

In this work, we followed the formulation proposed in Shu et al. (2020) where continuous latent variables $\boldsymbol{z} \in \mathbb{R}^{T' \times D}$ are modeled as spherical Gaussian at the encoder output of each position. Like typical VAEs (Kingma and Welling, 2013), the model is trained by maximizing the evidence lower-bound (ELBO) with a posterior network q_ϕ:

$$\underbrace{\mathop{\mathbb{E}}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x})\right]}_{\text{likelihood}} - \mathcal{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \| p_\theta(\boldsymbol{z}|\boldsymbol{x})) \quad (3)$$

where $\mathcal{D}_{\mathrm{KL}}$ is the Kullback–Leibler divergence between the prior and posterior. In this work, we use a Transformer to encode $q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$. Only the embedding layers are shared between θ and ϕ

3.3 Loss Function: Latent Alignments

Standard NMT models are trained with the cross entropy (CE) loss which compares the model's output with target tokens at each corresponded position. However, as NAT ignores the dependency in the output space, it is almost impossible for such models to model token offset accurately. For instance, while with little effect to the meaning, simply changing "*Vielen Dank !*" to "*, Vielen Dank*" causes a huge penalty for fully NAT models.

To ease such limitation, recent works proposed to consider the latent alignments between the target positions, and optimize (Ghazvininejad et al.,

Methods	Distillation	Latent Variables	Latent Alignments	Glancing Targets
What it can do?	simplifying the training data	model any types of dependency in theory	handling token shifts in the output space	ease the difficulty of learning hard examples
What it cannot?	uncertainty exists in the teacher model	constrained by the modeling power of the used latent variables	unable to model non-monotonic dependency, e.g. reordering	training / testing phase mismatch
Potential issues	sub-optimal due to the teacher's capacity	difficult to train; posterior collapse	decoder inputs must be longer than targets	difficult to find the optimal masking ratio

Table 1: Comparison between the proposed techniques for improving fully NAT models.

2020a), or marginalize all alignments (Libovický and Helcl, 2018; Saharia et al., 2020). As a special form of latent variables in loss computation, latent alignments can be easily computed through dynamic programming. The dependency is reduced because the NAT model is able to freely choose the best prediction regardless of the target offsets. In this work, we put our primary focus on Connectionist Temporal Classification (CTC) (Graves et al., 2006) as the latent alignments, considering its superior performance and the flexibility of variable length prediction. Formally, CTC is capable of efficiently finding all valid aligned sequences $\boldsymbol{a}$ which the target $\boldsymbol{y}$ can be recovered from, and marginalize log-likelihood:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \log \sum_{\boldsymbol{a}\in\Gamma(\boldsymbol{y})} p_\theta(\boldsymbol{a}|\boldsymbol{x}) \tag{4}$$

where $\Gamma^{-1}(\boldsymbol{a})$ is the collapse function that recovers the target sequence by collapsing consecutive repeated tokens, and then removing all blank tokens. Also, it is straightforward to apply the same CTC loss into the VAE models (§ 3.2) by replacing the likelihood term in Eq (3) with the CTC loss. Because of the strong assumptions of monotonic alignment, it is impossible to reduce all dependencies between target tokens in real distribution.

3.4 Learning: Glancing Targets

Ghazvininejad et al. (2019) showed that it improved test time performance by glancing the reference tokens when training NAT models. That is, instead of $\log p_\theta(\boldsymbol{y}|\boldsymbol{x})$, we optimize $\log p_\theta(\boldsymbol{y}|\boldsymbol{m}\odot \boldsymbol{y},\boldsymbol{x}), \boldsymbol{m} \sim \gamma(l,\boldsymbol{y}), l \sim \mathcal{U}_{|\boldsymbol{y}|}$, where $\boldsymbol{m}$ is the mask, and γ is the sampling function given the number of masked tokens l. As mentioned earlier, we suspect such explicit modeling of the distribution conditional to unmasked tokens assists the *dependency reduction* in the output space.

Naively applying random masks for every training example may cause severe mismatch between training and testing. To migrate this, Qian et al. (2020) proposed GLAT – a curriculum learning strategy, in which the ratio of glanced target tokens is proportional to the translation error of the fully NAT model. More precisely, instead of sampling uniformly, we sample l by:

$$l \sim g(f_{\text{ratio}} \cdot \mathcal{D}(\hat{\boldsymbol{y}},\boldsymbol{y})) \tag{5}$$

where $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \log p_\theta(\boldsymbol{y}|\boldsymbol{x})$, $\mathcal{D}$ is the discrepancy between the model prediction and the target sequence, e.g. Levenshtein distance (Levenshtein, 1966), and f_{ratio} is a hyperparameter to adjust the mask ratio. The original formulation (Qian et al., 2020) utilized a deterministic mapping (g), while we use a Poisson distribution to sample a wider range of lengths including "no glancing".

The original GLAT (Qian et al., 2020) assumes to work with golden length so that it can glance at the target by placing the target word embedding to the corresponded inputs, which is incompatible with CTC as we always require the inputs longer than the targets. To enable GLAT training, we glance at target tokens from the viterbi aligned tokens $\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}\in\Gamma(\boldsymbol{y})} \log p_\theta(\boldsymbol{a}|\boldsymbol{x})$ which has the same length as the decoder inputs. Intuitively, a poorly trained model will glance at many target tokens. When the model becomes better and generates higher quality sequences, the number of masked words will be larger, which helps the model gradually learn generating the whole sentence.

4 Experiments

We perform extensive experiments on three challenging translation datasets by combining all mentioned techniques to check (1) whether the proposed aspects for *dependency reduction* are complementary; (2) how much we can minimize the gap between a fully non-autoregressive model with the autoregressive counterpart.

4.1 Experimental Setup

Dataset and Preprocessing We validate our proposed models on three standard translation benchmarks with variant sizes, i.e., WMT14 English (EN) $\leftrightarrow$ German (DE) (4.0M pairs), WMT16 English (EN) $\leftrightarrow$ Romanian (RO) (610k pairs) and WMT20 Japanese (JA) $\rightarrow$ English (EN) (13M pairs after filtering). For EN$\leftrightarrow$DE and EN$\leftrightarrow$RO, we apply the same prepossessing steps and learn subwords as mentioned in prior work (EN$\leftrightarrow$DE: Zhou et al., 2020, EN$\leftrightarrow$RO: Lee et al., 2018). For JA$\rightarrow$EN, the original data (16M pairs) is first filtered with Bicleaner (Sánchez-Cartagena et al.) [2] and we apply SentencePiece (Kudo and Richardson, 2018) to generate 32,000 subwords.

Knowledge Distillation Following previous efforts, we also train the NAT models on distilled data generated from pre-trained transformer models (*base* for WMT14 EN$\leftrightarrow$DE and WMT16 EN$\leftrightarrow$RO and *big* for WMT20 JA$\rightarrow$EN) using beam search with a beam size 5 and length penalty 1.0.

Decoding At inference time, the most straightforward way is to generate the sequence with the highest probability at each position. The outputs from the CTC-based NAT models require an additional collapse process Γ^{-1} which can be done instantly. A relatively more accurate method is to decode multiple sequences, and rescore them to obtain the best candidate in parallel, i.e. *noisy parallel decoding* (NPD, Gu et al., 2018a). Furthermore, CTC-based models are also capable of decoding sequences using beam-search (Libovický and Helcl, 2018), and optionally combined with n-gram language models (Heafield, 2011; Kasner et al., 2020). More precisely, we search in a beam to approximately find the optimal $\boldsymbol{y}^*$ that maximizes:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) + \alpha \cdot \log p_{\text{LM}}(\boldsymbol{y}) + \beta \log |\boldsymbol{y}| \quad (6)$$

where α and β are hyperparameters for language model scores and word insertion bonus. In principle, it is no longer non-autoregressive as beam-search is a sequential process by nature. However, it does not contain any neural network computations and can be implemented efficiently in C++ [3].

Baselines We adopt Transformer (AT) and existing NAT approaches (see Table 2) for comparison. For AT, except for the standard *base* and *big* architectures (Vaswani et al., 2017), we also compare with a *deep encoder, shallow decoder* Transformer suggested in Kasai et al. (2020b) that follows the model dimensions of *base* with 12 encoder layers and 1 decoder layer (i.e. *base* (12-1) for short).

Evaluation BLEU (Papineni et al., 2002) is used to evaluate the translation performance for all models. Following prior works, we compute tokenized BLEUs for EN$\leftrightarrow$DE and EN$\leftrightarrow$RO, while using SacreBLEU (Post, 2018) for JA$\rightarrow$EN. In this work, we use three measures to fully investigate the translation latency of all the models:

- $\mathcal{L}_1^{\text{GPU}}$: translation latency by running the model with one sentence/batch on single GPU, aligning applications like instantaneous translation.
- $\mathcal{L}_1^{\text{CPU}}$: the same as $\mathcal{L}_1^{\text{GPU}}$ while running the model without GPU speed-up. Compared to $\mathcal{L}_1^{\text{GPU}}$, it is less friendly to NAT models that make use of parallelism, however, closer to real scenarios.
- $\mathcal{L}_{\max}^{\text{GPU}}$: the same as $\mathcal{L}_1^{\text{GPU}}$ on GPU while running the model in a batch with as many sentences as possible. In this case, the hardware memory bandwidth are taken into account.

We measure the wall-clock time for translating the whole test set, and report the averaged time over sentences as the latency measure. For more implementation details, please refer to Appendix A.

4.2 Results

WMT'14 EN$\leftrightarrow$DE & WMT'16 EN$\leftrightarrow$RO We report the performance of our fully NAT model comparing with AT and existing NAT approaches (including both iterative and fully NAT models) in Table 2. Iterative NAT models with enough number of iterations generally outperform fully NAT baselines by a certain margin as they are able to recover the generation errors by explicitly modeling dependencies between (partially) generated tokens. However, the speed advantage is relatively small compared to AT *base* (12-1) which also achieves 2.5 times faster than the AT baseline.

Conversely, our fully NAT models are able to readily achieve over 16 times speed-up on EN$\rightarrow$DE by restricting translation within a single iteration. Surprisingly, merely training NAT with KD and CTC loss already beats the state-of-the-art for single iteration NAT models across all four directions. Moreover, combining with either latent variables (VAE) or glancing targets (GLAT) further closes the performance gap or even outperforms the AT results on both language pairs. For example, our best

[2] https://github.com/bitextor/bicleaner
[3] https://github.com/parlance/ctcdecode

Models		Iter.	Speed	WMT'14 EN-DE	WMT'14 DE-EN	WMT'16 EN-RO	WMT'16 RO-EN
AT	Transformer *base* (teacher)	N	1.0×	**27.48**	**31.39**	**33.70**	**34.05**
	Transformer *base* (12-1)	N	2.4×	26.21	30.80	33.17	33.21
	+ KD	N	2.5×	27.34	30.95	33.52	34.01
Iterative NAT	iNAT (Lee et al., 2018)	10	1.5×	21.61	25.48	29.32	30.19
	Blockwise (Stern et al., 2018)	$\approx N/5$	3.0×	27.40	-	-	-
	InsT (Stern et al., 2019)	$\approx \log N$	4.8×	27.41	-	-	
	CMLM (Ghazvininejad et al., 2019)*	10	1.7×	27.03	30.53	33.08	33.31
	LevT (Gu et al., 2019)	Adv.	4.0×	27.27	-	-	33.26
	KERMIT (Chan et al., 2019)	$\approx \log N$	-	27.80	30.70	-	-
	LaNMT (Shu et al., 2020)	4	5.7×	26.30	-	-	29.10
	SMART (Ghazvininejad et al., 2020b)*	10	1.7×	27.65	31.27	-	-
	DisCO (Kasai et al., 2020a)*	Adv.	3.5×	27.34	31.31	33.22	33.25
	Imputer (Saharia et al., 2020)*	8	3.9×	**28.20**	**31.80**	**34.40**	**34.10**
Fully NAT	Vanilla-NAT (Gu et al., 2018a)	1	15.6×	17.69	21.47	27.29	29.06
	LT (Kaiser et al., 2018)	1	3.4×	19.80	-	-	-
	CTC (Libovický and Helcl, 2018)	1	-	16.56	18.64	19.54	24.67
	NAT-REG (Wang et al., 2019)	1	-	20.65	24.77	-	-
	Bag-of-ngrams (Shao et al., 2020)	1	10.0×	20.90	24.60	28.30	29.30
	Hint-NAT (Li et al., 2018)	1	-	21.11	25.24	-	-
	DCRF (Sun et al., 2019)	1	10.4×	23.44	27.22	-	-
	Flowseq (Ma et al., 2019)	1	1.1 ×	23.72	28.39	29.73	30.72
	ReorderNAT (Ran et al., 2019)	1	16.1×	22.79	27.28	29.30	29.50
	AXE (Ghazvininejad et al., 2020a)*	1	15.3×	23.53	27.90	30.75	31.54
	ENGINE (Tu et al., 2020)	1	15.3×	22.15	-	-	33.16
	EM+ODD (Sun and Yang, 2020)	1	16.4×	24.54	27.93	-	-
	GLAT (Qian et al., 2020)	1	15.3×	25.21	29.84	31.19	32.04
	Imputer (Saharia et al., 2020)*	1	18.6×	25.80	28.40	32.30	31.70
	Ours (Fully NAT)	1	17.6×	11.40	16.47	24.52	24.79
	+ KD	1	17.6×	19.50	24.95	29.91	30.25
	+ KD + CTC	1	16.8×	26.51	30.46	33.41	34.07
	+ KD + CTC + VAE	1	16.5×	**27.49**	31.10	**33.79**	33.87
	+ KD + CTC + GLAT	1	16.8×	27.20	**31.39**	33.71	**34.16**

Table 2: Comparison between our models and existing methods. The speed-up is measured on WMT'14 En→De test set. All results reported standalone are without re-scoring. **Iter.** denotes the number of iterations at inference time, **Adv.** means adaptive, * denotes models trained with distillation from a *big* Transformer.

model achieves **27.49** BLEU on WMT14 EN-DE – almost identical to the AT performance (27.48) while $\mathbf{16.5}$ times faster in the inference time.

Table 2 also indicates the difficulties of learning NAT on each dataset. For instance, EN↔RO is relatively easier as "KD + CTC" is enough to close the performance gap. By contrast, applying VAE or GLAT helps to capture non-monotonic dependencies and improve by $0.5 \sim 1$ BLEU points on EN↔ED. For both datasets, we ONLY need a single greedy generation to achieve similar translation quality as AT beam-search results.

WMT'20 JA→EN In Table 3, we also present results for training the fully NAT model on a more challenging benchmark – WMT'20 JA→EN which is much larger (13M pairs) and noisier. In addition, JA is linguistically distinct from EN which makes it harder to learn mappings between them. Consequently, both AT (12-1) and our fully NAT models become less confident and tend to generate shorter translations (BP <0.9), which in turn underperform the AT teacher even trained with KD.

Beam search & NPD Previous works (Gu et al., 2018a; Libovický and Helcl, 2018) find that NAT performance can be effectively improved by allowing advanced decoding methods, such as beam-search and re-ranking (NPD). To fully examine our proposed fully NAT model and demonstrate its extensibility with advanced decoding approaches, we further conduct experiments on WMT'20 JA→EN.

For CTC beam search, we use a fixed beam-size 20 while grid-search α, β (Eq.(6)) based on the performance on the validation set. The language model [4] is trained directly on the distilled target sentences to avoid introducing additional information. We explored both 3-gram and 4-gram LMs in our initial experiments, and found 4-gram worked slightly better with no effect on the infer-

[4] https://github.com/kpu/kenlm

	Configuration	BLEU (Δ)	BP	$\mathcal{L}_1^{\text{GPU}}$ (Speed-up)		$\mathcal{L}_1^{\text{CPU}}$ (Speed-up)	
AT	*big* (teacher)	21.07	0.920	345 ms	1.0 ×	923 ms	1.0 ×
	base	18.91	0.908	342 ms	1.0 ×	653 ms	1.4 ×
	base (12-1)	15.47	0.806	152 ms	2.3 ×	226 ms	4.0 ×
	base (12-1) + KD	18.76	0.887	145 ms	2.4 ×	254 ms	3.6 ×
NAT	KD + CTC	16.93 (+0.00)	0.828	17.3 ms	19.9 ×	84 ms	11.0 ×
	KD + CTC + VAE	18.73 (+1.80)	0.862	16.4 ms	21.0 ×	83 ms	11.1 ×
	w. *BeamSearch20*	19.80 (+2.87)	0.958	28.5 ms	12.1 ×	99 ms	9.3 ×
	w. *BeamSearch20 + 4-gram LM*	**21.41 (+4.48)**	0.954	31.5 ms	11.0 ×	106 ms	8.7 ×
	w. *NPD5*	18.88 (+1.95)	0.866	34.9 ms	9.9 ×	313 ms	2.9 ×
	w. *NPD5 + BeamSearch20 + 4-gram LM*	**21.84 (+4.91)**	0.962	57.6 ms	6.0 ×	284 ms	3.2 ×

Table 3: Performance comparison between fully NAT and AT models on WMT'20 JA→EN. Translation latency on both the GPU and CPUs are reported over the test set. The brevity penalty (BP) is also shown for reference.

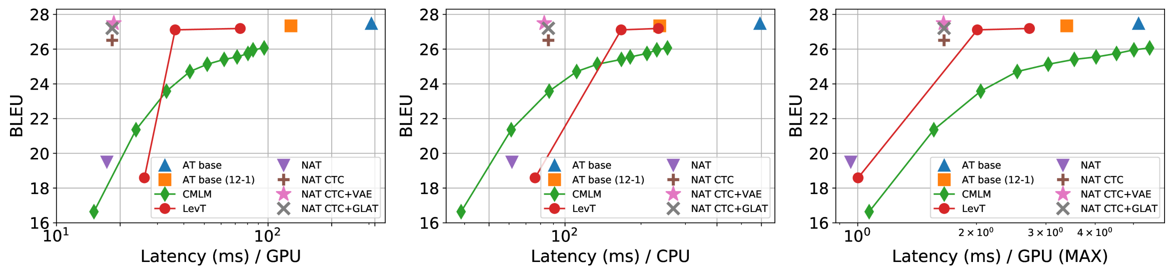

Figure 4: Quality v.s. Latency (the upper left corner achieves the best trade-off) for fully NAT models with other translation models (AT *base* and *base* 12-1 (Kasai et al., 2020b), CMLM (Ghazvininejad et al., 2019) and LevT (Gu et al., 2019)) on WMT'14 EN→DE. We evaluate latency in three setups (from left to right: $\mathcal{L}_1^{\text{GPU}}$, $\mathcal{L}_1^{\text{CPU}}$, $\mathcal{L}_{\text{max}}^{\text{GPU}}$) and show them in Logarithmic scale for better visualization.

ence speed. For noisy parallel decoding (NPD), we draw multiple z from the learned prior distribution with temperature 0.1, and use the teacher model to rerank the best z with the corresponded translation.

As shown in Table 3, with similar GPU latency ($\mathcal{L}_1^{\text{GPU}}$), beam search is much more effective than NPD with re-ranking, especially combined with a 4-gram LM where we achieve a BLEU score of 21.41, beating the teacher model with $11\times$ speed-up. More importantly, by contributing the insertion bonus (3rd term in Eq (6)) with β in beam search, we have the explicit control to improve BP and output longer translations. Also, we gain another half point by combining NPD and beam search. To have a fair comparison, we also report latency on CPUs where it is limited to leverage parallelism of the device. The speed advantage drops rapidly for NAT models, especially for NAT with NPD, however, we still maintain around 100 ms latency via beam search – over $2\times$ faster than the lightweight AT (12-1) systems with higher translation quality.

Quality v.s. Latency We perform a full investigation for the trade-off between translation quality and latency under three measures defined in § 4.1. The results are plotted in Figure 4. For fully NAT models, no beam search or NPD is considered. The latency is measured by $\mathcal{L}_1^{\text{GPU}}$, $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ so as to understand this trade-off in various scenarios. In all three setups, our fully NAT models obtain superior trade-off compared with AT and iterative NAT models. Iterative NAT models (LevT and CMLM) require multiple iterations to achieve reliable performance with the sacrifice of latency, especially under $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ where iterative NAT performs similarly or even worse than AT *base* (12-1), leaving fully NAT models a more suitable position in quality-latency trade-off.

Figure 4 also shows the speed advantage of fully NAT models shrinks in the setup of $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ where parallelism is constrained. Moreover, NAT models particularly those with CTC consume more computation and memory compared to AT models with a shallow decoder. For instance when calculating $\mathcal{L}_{\text{max}}^{\text{GPU}}$, we notice that the maximum allowed batch is 120K tokens for AT base (12-1), while we can only compute 15K tokens at a time for NAT with CTC due to the up-sampling step, even though the NAT models still win the wall-clock time. We

KD	AXE	CTC	VAE	RND	GLAT	BLEU
						11.40
✓						19.50
	✓					16.59
✓	✓					21.66
		✓				18.18
✓		✓				**26.51**
		✓	✓			23.58
✓	✓		✓			22.19
✓		✓	✓			**27.49**
✓	✓			✓		22.74
✓	✓				✓	24.67
✓		✓		✓		26.16
		✓			✓	21.81
✓		✓			✓	**27.20**
✓		✓	✓		✓	**27.21**

Table 4: Ablation on WMT'14 EN→DE test set with different combinations of techniques. The default setup shows a plain NAT model (Gu et al., 2018a) directly trained on raw targets with the cross entropy (CE) loss.

mark it as one limitation for future research.

4.3 Ablation Study

Impact of various techniques Our fully NAT models benefit from dependency reduction techniques in four aspects (data, model, loss function and learning), and we analyze their effects on translation accuracy through various combinations in Table 4. First of all, the combinations without KD have clear performance drop compared to those with KD, showing its vital importance in NAT training. For the loss function, although both AXE (Ghazvininejad et al., 2019) and CTC consider the latent alignments, the CTC-based model obtains much better accuracy due to its flexibility of output length. In all cases, incorporating latent variables also effectively improves the accuracy, especially for CTC without KD ($\sim$ 5 BLEU improvement). Because of the capability to reduce the mismatch between training and inference time, the model with GLAT is superior to those with randomly (RND) sampled masks. To conclude, we find that KD and CTC are necessary components for a robust fully NAT model. Adding either VAE or GLAT to them achieve similar improvements.

Distillation corpus We report the performance of models trained on real data and distilled data generated from AT *base* and *big* models in Table 5. For *base* models, both AT (12-1) and NAT achieve better accuracy with distillation, while AT benefits more by moving from *base* to *big* distilled data. On the contrary, the NAT model improves marginally indicating that in terms of the modeling capacity, our fully NAT model is still worse than AT model even with 1 decoder layer. It is not possible to further boost the NAT performance by simply switching the target to a better distillation corpus, which aligns the finding in Zhou et al. (2020). Nonetheless, we can increase the NAT capacity by learning in *big* size. As shown in Table 5, we can achieve superior accuracy compared to AT (12-1) with little effect on the translation latency ($\mathcal{L}_1^{\text{GPU}}$).

Models		Distillation *base*	Distillation *big*	BLEU	Speed-up
AT	*base*			27.43	1.0×
	big			28.14	0.9×
	base (12-1)			26.12	2.4×
		✓		27.34	2.5×
			✓	27.83	2.4×
NAT	*base*			23.58	16.5×
		✓		27.49	16.5×
			✓	27.56	16.5×
	big		✓	**27.89**	15.8×

Table 5: Performance comparison between AT and NAT models on the test set of WMT'14 EN→DE. The latency is measured one sentence per batch and compared with the Transformer *base*. For NAT model, we adopt CTC+VAE as the basic configuration.

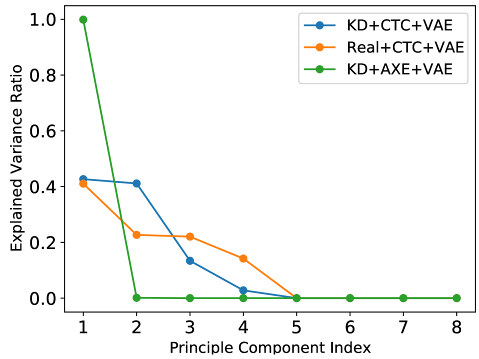

Figure 5: Principle component explained variance ratios of latent variables on WMT'14 EN→DE test set.

Effective Latent Dimensionality of Latent Variables To confirm the necessity of combining VAEs with CTC, We apply principal component analysis (PCA) (Wold et al., 1987) on the learned latent variables. More precisely, we extract the latent variables from the posterior of various models (see Table 4) on WMT'14 EN→DE test set. These main components' explained variance ratios, the percentage of variance that is attributed by each of the component, are shown in Figure 5.

First, we find that the number of effective latent dimensionality (capturing at least 95% of the total variance) is much lower than the number of latent dimensions (8 in our experiments), which indicates simply increasing the number of latent dimensions does not lead to better representations, and the ability to capture dependencies is limited. Therefore, VAEs need to be combined with other techniques e.g. KD, CTC to take effect. Also, compared to the AXE, the effective dimensionality of latent variables in CTC loss-based models is higher.

We include more analysis with qualitative examples in Appendix B.

5 Discussion and Future work

In this section, we go through the proposed four techniques again for fully NAT models. In spite of the success to close the gap with autoregressive models on certain benchmarks, we still see limitations when using non-autoregressive systems as mentioned in Table 1.

We and most of the prior research have repeatedly found that knowledge distillation (KD) is the indispensable *dependency reduction* components, especially for training fully NAT models. Nevertheless, we argue that due to the model agnostic property, KD may lose key information that is useful for the model to translate. Moreover, Anonymous (2021) pointed out KD does cause negative effects on lexical choice errors for low-frequency words in NAT models. Therefore, an alternative method that improves the training of NAT models over raw targets using such as GANs (Bińkowski et al., 2019) or domain specific discriminators (Donahue et al., 2020) might be the future direction.

Apart from KD, we also notice that the usage of CTC loss is another key component to boost the performance of fully NAT models across all datasets. As discussed in § 4.2, however, the need of up-sampling constrains the usage of our model on very long sequences or mobile devices with limited memory. In future work, it is possible to explore models to hierarchically up-sample the length with a dynamic ratio to optimize the memory usage.

Lastly, both experiments with VAE and GLAT prove that it is helpful but not enough to train NAT models with loss based on monotonic alignments (e.g. CTC) only. To work on difficult pairs such as JA-EN, it may be a better option to adopt stronger models to capture richer dependency information, such as normalizing flows (van den Oord et al., 2018; Ma et al., 2019) or non-parametric approaches (Gu et al., 2018b).

6 Related Work

Besides iterative NAT and fully NAT models, there are other works trying to improve the decoding speed of translation models from other aspects. One research line is to hybrid AT and NAT models. Wang et al. (2018) proposed a semi-autoregressive model which adopted non-autoregressive decoding locally but kept the autoregressive property in global. On the contrary, Kong et al. (2020); Huang et al. (2017) and Ran et al. (2020) introduced a local autoregressive NAT models which retained the non-autoregressive property in global.

Alternatively, there are also efforts improving the decoding speed of AT models directly. Model quantization and pruning have been widely studied as a way to improve the decoding speed (See et al., 2016; Junczys-Dowmunt et al., 2018; Aji and Heafield, 2020). Also, specialized light-weight AT model (e.g. replacing self-attention with SSRU) together with improved teacher-student training (Kim et al., 2019) are explored.

7 Conclusion

In this work, we aim to minimize the performance gap between fully NAT and AT models. We investigate *dependency reduction* methods from four perspectives and carefully unite them with necessary revisions. Experiments on three translation benchmarks demonstrate that the proposed fully NAT models achieve the SoTA performance. For future work, it is worth exploring simpler but more effective diagrams for learning NAT models. For instance, with the combination of CTC and more powerful latent variable models, it is possible to remove the necessity of knowledge distillation.

Acknowledgements

We would like to thank Jason Lee, Xuezhe Ma and Chunting Zhou for thoughtful discussion. We would also like to thank the anonymous reviewers for their time and providing helpful suggestions.

References

Alham Fikri Aji and Kenneth Heafield. 2020. Compressing neural machine translation models with 4-bit precision. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 35–42, Online. Association for Computational Linguistics.

Anonymous. 2021. Understanding and improving lexical choice in non-autoregressive translation. In *Submitted to International Conference on Learning Representations*. Under review.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C Cobo, and Karen Simonyan. 2019. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. 2020. End-to-end adversarial text-to-speech. *arXiv preprint arXiv:2006.03575*.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3515–3523. PMLR.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, Canada, April 30-May 3, 2018, Conference Track Proceedings*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11181–11191. Curran Associates, Inc.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018b. Search engine guided neural machine translation.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2017. Towards neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018. Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 129–135, Melbourne, Australia. Association for Computational Linguistics.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2395–2404.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020a. Non-autoregressive machine translation with disentangled context transformer. In *International Conference on Machine Learning*, pages 5144–5155. PMLR.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. 2020b. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *arXiv preprint arXiv:2006.10369*.

Zdeněk Kasner, Jindřich Libovickỳ, and Jindřich Helcl. 2020. Improving fluency of non-autoregressive machine translation. *arXiv preprint arXiv:2004.03227*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1067–1073, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1173–1182.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Hint-based training for non-autoregressive translation.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. 2018. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926, Stockholmsmässan, Stockholm Sweden. PMLR.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2020. Glancing transformer for non-autoregressive neural machine translation. *arXiv preprint arXiv:2008.07905*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3059–3069, Online. Association for Computational Linguistics.

Yi Ren, Jinglin Liu, Xu Tan, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. *arXiv preprint arXiv:2004.10454*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

Víctor M. Sánchez-Cartagena, Marta Bañón, Sergio Ortiz-Rojas, and Gema Ramírez-Sánchez. Prompsit's submission to wmt 2018 parallel corpus filtering shared task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared*

Task Papers, Brussels, Belgium. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 198–205.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985, Long Beach, California, USA. PMLR.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, pages 10107–10116.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, pages 3016–3026.

Zhiqing Sun and Yiming Yang. 2020. An em approach to non-autoregressive conditional sequence generation. In *International Conference on Machine Learning*, pages 9249–9258. PMLR.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5377–5384.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, Jun Xie, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1906.02041*.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, Conference Track Proceedings*.

Appendix

A Implementation Details

Architecture We design our fully NAT model with the hyperparameters of the *base* Transformer: 8-512-2048 (Vaswani et al., 2017). For EN→DE experiments, we also implement the NAT model in *big* size: 8-1024-4096 for comparison.

VAEs For experiments using variational autoencoders (VAE), we use the last layer encoder hidden states to predict the mean and variance of the prior distribution. The latent dimension D is set to 8, and the predicted $\boldsymbol{z}$ are linearly projected and added on the encoder outputs. Following Shu et al. (2020), we use a 3 layer encoder-decoder as the posterior network, and apply freebits annealing (Chen et al., 2016) to avoid posterior collapse.

CTC By default, we upsample the length of decoder inputs $3\times$ as long as the source for CTC, while using the golden length for other objectives (CE and AXE). We also train an additional length predictor when CTC is not used. For both cases, we use *SoftCopy* (Wei et al., 2019) which interpolated the encoder outputs as the decoder inputs based on the relative distance of source and target positions.

GLAT The mask ratio, f_{ratio}, is 0.5 for GLAT training. The original GLAT (Qian et al., 2020) assumes to work with the golden length so that it can glance at the target by placing the target word embedding to a clear corresponded inputs. It is incompatible with CTC loss where we always need longer inputs than the targets. To enable GLAT learning, we glance at target tokens from the viterbi aligned tokens ($\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}\in\beta(\boldsymbol{y})} p(\boldsymbol{\alpha}|\boldsymbol{x})$) which has the same length as the decoder inputs.

Training For both AT and NAT models, we set the dropout rate as 0.3 for EN↔DE and EN↔RO, and 0.1 for JA→EN. We apply weight decay 0.01 as well as label smoothing $\epsilon = 0.01$. All models are trained for 300K updates using Nvidia V100 GPUs with a batch size of approximately 128K tokens. We measure the validation BLEU scores for every 1000 updates, and average the last 5 checkpoints to obtain the final model.

Inference We measure the GPU latency by running the model on a single Nvidia V100 GPU, and CPU latency on Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz with 80 cores. All models are implemented on `fairseq` (Ott et al., 2019).

λ	BLEU	$\mathcal{L}_1^{\text{GPU}}$	$\mathcal{L}_{\max}^{\text{GPU}}$	$\mathcal{L}_1^{\text{CPU}}$
1.5	26.16	17.9 ms	**0.95 ms**	**66.6 ms**
2.0	26.39	17.5 ms	1.03 ms	71.6 ms
2.5	**26.54**	17.6 ms	1.16 ms	76.9 ms
3.0	26.51	**17.0 ms**	1.32 ms	81.8 ms

Table 6: Performance comparison of different upsample ratios (λ) for CTC-based models on WMT'14 EN→DE test set. All models are trained on distilled data.

B More ablation study

Upsampling Ratio (λ) for CTC Loss To meet the length requirements in CTC loss, we upsample the encoder output by a factor of 3 in our experiments. We also explore other possible values and report the performance in Table 6. The higher upsampling ratio provides a larger alignment space, leading to better accuracy. Nevertheless, with a large enough sampling ratio, a further increase will not lead to the performance increase. Because of the high degree of parallelism, $\mathcal{L}_1^{\text{GPU}}$ speed is similar among these ratios. However, the model with a larger ratio has a clear latency drop on CPU or GPU with large batches.

Representation reordering in the latent space In our main experiments, VAEs has been proven to effectively improve the performance of NAT models. Here, we perform a qualitative study to show how VAEs helps NAT models.

Ott et al. (2018) collected additional reference translations for each source sentence in the WMT'14 En→De test set. We first choose three source sentences and show the alignments between them and two of their different translations in Figure 6. In each sample, it is clear to find that the word order of the first pair is more similar to the second one (e.g., in the second sample, the verb 'light' in the source sentence is translated to the end of the second reference sentence). However, given the monotonic alignment assumption, CTC is difficult to align sentence pairs with different word orders. Then, for each sample, we extract latent variables of both sentence pairs and align them by first computing the Euclidean distance between every position and then employing the linear sum assignment algorithm (LAP).

Regarding the first pair as the baseline, we find that the latent variable is able to adjust the word order according to the input sentence pair. For example, the alignment between latent variables of

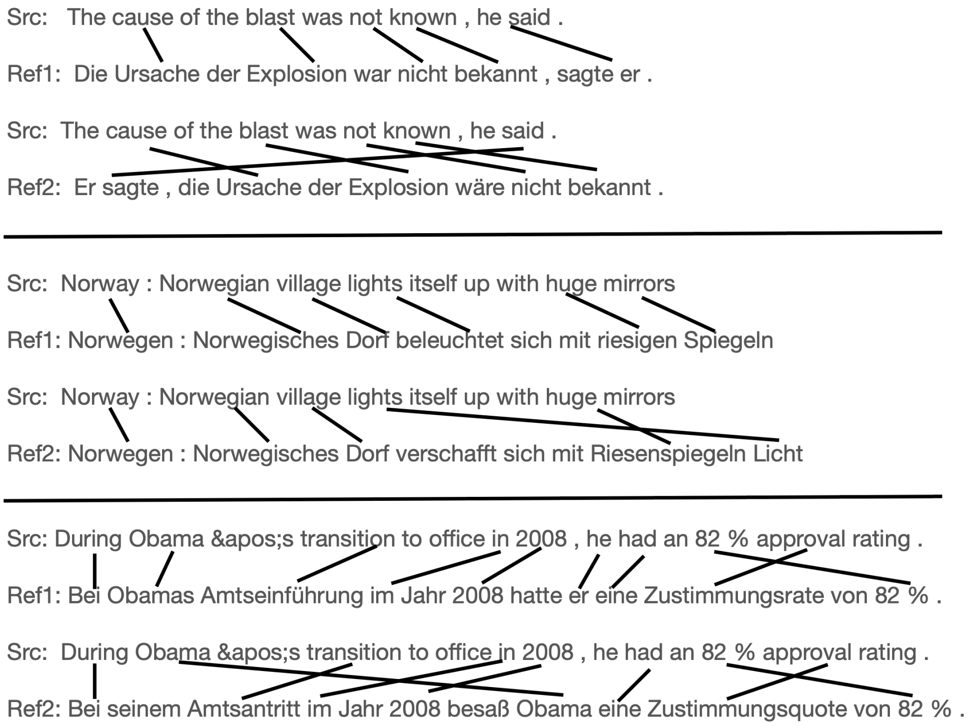

Figure 6: Alignments between source sentences and their different translations.

the second sample is shown as: 0-0, 1-1, 2-2, 3-3, 4-9, 5-5, 6-6,7-7, 8-8, 9-4, which shows that the latent representation of the 9th position in the second pair is aligned to the 5th position of the second pair. In another word, the latent representation of the word 'lights' is reordered to the last position in the second pair's latent variable, which corresponds to the word order difference in the second pair. Therefore, given various reference information, the latent variable makes the alignment between the source and target representation more monotonic. CTC can consequently benefit from it to learn a better NAT model.